Proceedings of the 29th Annual International
Conference of the IEEE EMBS
Cité Internationale, Lyon, France
August 23-26, 2007.

A New Model for Programming Software in Body Sensor Networks

Talles M. G. de A. Barbosa, Iwens G. Sene Jr, Adson F. da Rocha, Francisco A. de O. Nascimento, Joao L. A. Carvalho and Hervaldo S. Carvalho

Abstract — **A Body Sensor Network (BSN) must be designed to work autonomously. On the other hand, BSNs need mechanisms that allow changes in their behavior in order to become a clinically useful tool. The purpose of this paper is to present a new programming model that will be useful for programming BSN sensor nodes. This model is based on an intelligent intermediate-level compiler. The main purpose of the proposed compiler is to increase the efficiency in system use, and to increase the lifetime of the application, considering its requirements, hardware possibilities and specialist knowledge. With this model, it is possible to maintain the autonomous operation capability of the BSN and still offer tools that allow users with little grasp on programming techniques to program these systems.**

Keywords— **Body Sensor Network (BSN), deployment-time programmability, application-domain requirements, transparency.**

I. Introduction

Pervasive monitoring demands great adaptation capability of the Body Sensor Networks (BSNs) [1]. In BSNs, cases in which the decision made by the system is different from the decision that would have been made by a healthcare professional are frequent. In order to become a clinically useful tool, BSNs need intelligent algorithms for autonomous operation, and mechanisms that allow changes in their behavior. According to Baldus et al. [2], "the BSN has to work automatically, but has also to be always under explicit control of any clinician".

This paper introduces a deployment-time programming model that could be useful for people with little grasp on programming techniques. Deployment-time programmability refers to the definition of software artifacts and algorithms that are embedded in the sensor nodes [3,4]. The inclusion of this functionality in BSNs requires a programming interface that is suitable for healthcare personnel, and intelligent compilers. Such compilers should be capable of handling implicit functional and non-functional requisites of a program. Deployment-time programmability could be useful for adding mechanisms and policies for energy saving to the BSN, for example.

Manuscript received April 2, 2007.

T. M. G. de A. Barbosa and I. G. Sene Jr. are with the Department of Computing, Catholic University of Goias, Goiania, GO 74605-110 Brazil (e-mail:talles@ucg.br, iwens@ucg.br).

A. F. da Rocha and F. A. de O. Nascimento are with the Department of Electrical Engineering, University of Brasília, Brasília, DF 70910-900 Brazil (e-mail: adson@unb.br, assis@unb.br).

J. L. A. Carvalho is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564 USA (e-mail: jcarvalh@usc.edu).

H. S. Carvalho is with the Medical School and with the Department of Electrical Engineering, University of Brasília, Brasília, DF 70910-900 Brazil (e-mail: carvalho@unb.br).

BSN research usually uses a bottom-up approach, where the sensor network hardware is usually chosen based on its availability. Software engineering abstractions are used in order to avoid low-level hardware issues, usually by including an intermediate-level compiler between the high-level programming model and the hardware programming tools. For example, BeanWatcher [5] allows the generation of applications based on a visual description of the components. Abstract Regions [6] offers an Application Programming Interface (API) to hide details of the radio transmitter, the routing protocols and the data sharing programming. VM* [7] is a framework that has been developed to allow application programming using Java syntax. In the ATaG [8], application programming is done using two graphs: the "abstract task graph" is responsible for defining the functionalities of the applications, and the "network graph" describes the network topology. These two information sets are processed by an intermediate-level compiler, which produces application source codes that may be compiled into binary code using traditional compilers like *gcc*. While being able to increase the transparency levels between the hardware and the programmers, such solutions assume that the BSN user has programming language skills. Moreover, these solutions do not take application-domain requirements and organization into consideration.

The solution presented in this paper addresses these issues, and is based on an intelligent intermediate-level compiler. An intelligent compiler is capable of defining mechanisms and adjusting policies that will compose the system software image deployed into sensor nodes. Its main purpose is to increase the efficiency in system use, therefore increasing the lifetime of the application. Such compilers should consider application domain requirements, and hardware possibilities. The specialist's knowledge should also be taken into consideration, as the medical expert could help in scheduling resources according to each application. The proposed model can increase the BSNs capability for autonomous operation, and offer tools that allow people with little grasp on programming techniques to program these systems.

This work is organized as follows. First, the description of the new model for BSN software programming is presented. Next, we show that it is possible to increase the lifetime of the applications by taking into account some hardware characteristics, associated with application-domain requirements and the specialist's knowledge during the sensor node programming. In conclusion, we propose this model as a framework for future research.

II. A New Model For Programming Software in Body Sensor Networks

An overview of the activities defined by the BSN programming model is presented in Fig. 1. The core of this model is an intermediate-level compiler, called agent generator. The ***functionality selection*** (shown in number 1) refers to the description of mechanisms and policies that will compose an application. This activity may be executed by any clinician through the graphical user programming interface. Next, the ***applications requirements are sent to*** the ***agent generator.*** The agent generator is composed of three modules: *(i)* Parser Module, *(ii)* Selector Module and *(iii)* Generator Module. The "Parser" module is responsible for extracting relevant information for the "Selector" module. In practice, these informations are related to parameters, such as the objective of the application, the number of tasks, the types of tasks, and the selected alarms, among others. The values of these parameters are used for filling a table of parameters that will be used by the selector module. The "Selector" module chooses the agent that maximizes the potential of each application, guaranteeing a longer lifetime for the system. In practice, the selector module is a decision tree [9] that is responsible for the choice of the best agent, from a set of possible agents, called agent family, that are deposited in an agent repository. This choice is made based on the table of values that is made available by the "Parser" module. The definition of this tree (selection algorithm) is based on the medical knowledge that influences the values of each parameter in the table. As an example, Fig. 2a shows an ECG agent family while Fig. 2b shows an ECG agent selected from that family.

After choosing the best agent, the "Generator" module generates the "agent.c" file, which is the application source code. A description of the code libraries and system calls that are needed should be used for this task. This description is stored in the agent repository, along with the functional description of each agent. During the agent selection, this configuration file is used by the "Generator" module for assembling the "agent.c" file. After that, ***the compilation and deployment in the target platform*** is executed (third activity). In order to do so, traditional compilers like *gcc* should be used. In this project, the MSPGCC [10], a tool for Texas Instruments MSP430 microcontrollers, was used. The

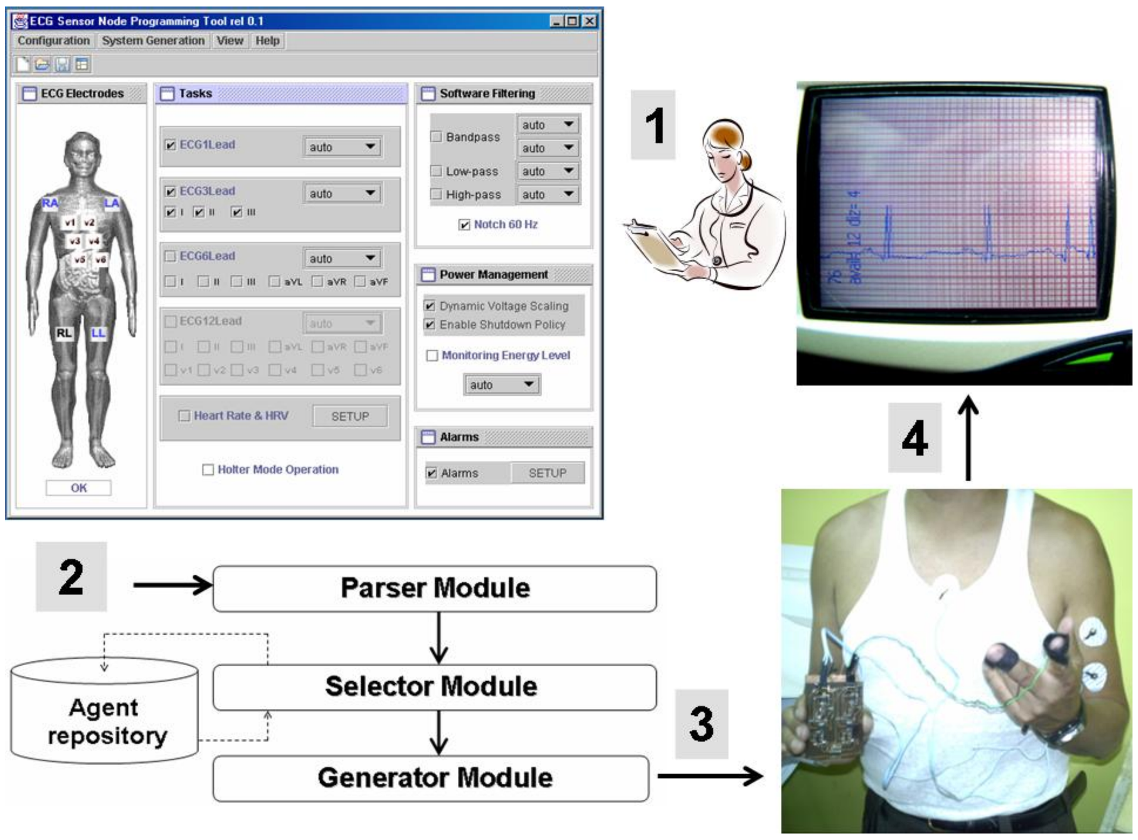

Fig.1. The BSN programming model activities centered in the intelligent compiler.

final activity (number 4 in the Fig. 1) is the ***system initialization***, which provides the monitored signals to the healthcare professional. In the example in Fig 1, a healthcare assistant evaluates an ECG signal, which is being displayed in a cell phone screen.

An ECG agent family is shown in Fig. 2a. This representation shows one set of tradeoff (tasks scheduling) possibilities for the ECG system, according to the specialist's knowledge. Specifically for these automata, tradeoff events may be influenced by noise characteristics, depending on the number of ECG leads, and the sampling rate. Noise analysis may or may not take into account the needs of the ECG diagnosis process. In fact, there are many other configuration possibilities for an ECG agent family that were not considered in this picture.

Each sub-set of an agent family may be considered an agent. It represents a piece of the specialist's knowledge delimited by specific application requirements captured during the functionalities selection stage. Fig 2b shows a selected ECG agent based on the functionality selection showed in the graphical programming interface screen presented in Fig 1.

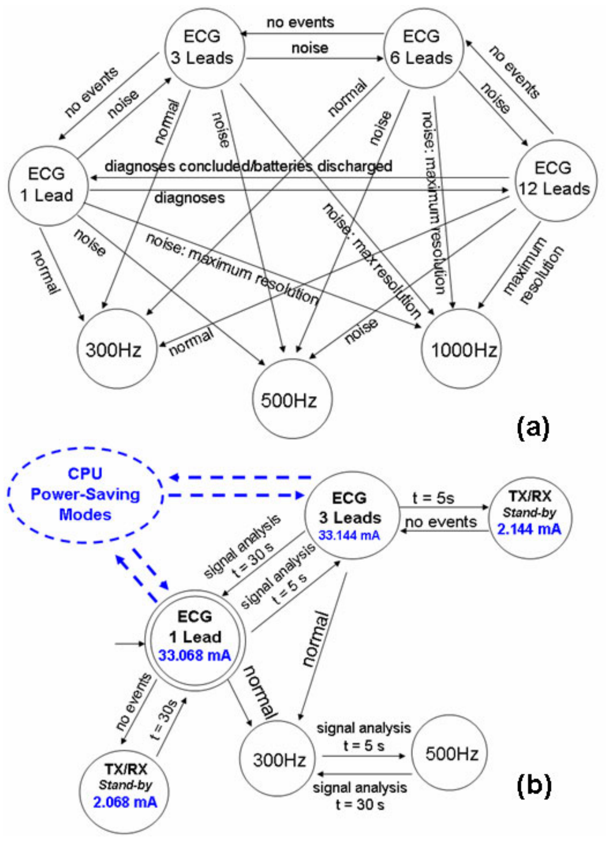

Fig. 2. (a) An ECG agent family. (b) An ECG agent selected from (a).

Energy saving can be obtained by using Dynamic Power Management (DPM) concepts, whereby the sensor node is shut down if no interesting event occur, or slowed down during periods of reduced activity [11]. It is desirable that the node has graceful energy quality scalability so that, if the application demands, the user is able to extend the mission lifetime at the cost of sensing accuracy.

In order to allow DPM appliance in the BSNs, application needs should be taken into account, otherwise an efficient "graceful degradation" can not be reached. Moreover, systems should consider the hardware possibilities and the specialist's knowledge, since the medical expert could help in scheduling resources according to each application.

The selected agent presented in Fig 2b represents an application that was programmed to support only two functionalities: the ECG 1 and the ECG 3 Leads. Mechanisms and policies were associated to each functionality, based on the specialist's knowledge, in order to promote the energy saving. As a consequence, the application lifetime will be extended in a elegant and efficient way. Mechanisms allow dynamic changes in hardware configuration, while policies are responsible for determining when these changes should occur.

III THE CPU POWER-SAVING MODES PERFORMANCE

In this work, we extend the DPM feature presented in a previous work [3] by taking into account specific characteristics of the hardware used in this project. In particular, we investigated the performance of the new energy saving possibilities when changes in the CPU Power-Saving modes occur (see Fig 2b).

Our hardware system, presented in Fig. 1, was assembled based on the Olimex MSP430-P149 kit [12], with a bluetooth radio BlueSMiRF Basic [13]. The ECG acquisition system was built based on the differential amplifier (Texas Instruments, USA).

The TI MSP430 MCUs may be configured to support dynamic changes in the operating modes. The processor has five low power modes: active mode, LPM0, LMP1, LMP2, LPM3 and LPM4. In each mode, the MSP430 has the option of shutting down the processor portion of the device, by using the CPUOff bit in the Status Register. As the current consumption increases with the clock frequency, the MSP430 also allows dynamic frequency adjusting in order to save energy.

Experiments to quantify the current consumption have been performed to evaluate the energy consumption of some possible configurations of the ECG sensor node. An overview of the results is presented in Fig. 3. The frequency range was chosen such that these frequencies would not shut down other MCU parts, like the A/D converter and the USART ports.

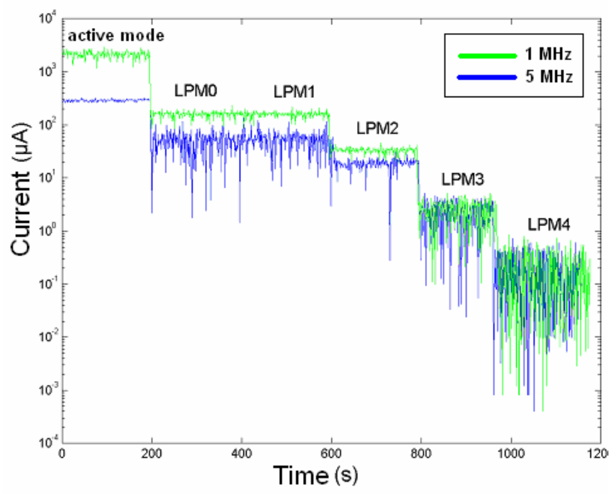

Fig 3. The performance of the CPU Power-Saving Modes. This picture represents the current consumption obtained from the ECG sensor node CPU when dynamic configurations in the operating modes and in the frequency operation were executed.

IV Discussion

Although the energy saving obtained from the dynamic changes in CPU modes are not as high as the savings obtained by when the bluetooth radio transmitter is in stand-by (see Fig 2b), the results presented in Fig 3 reveal that there are many mechanisms that can be used for improving energy saving in BSN, when hardware possibilities are also considered. In order to manage these mechanisms in an efficient and elegant way, we proposed a solution that is executed during the programming stage.

IV Conclusions and Future Works

In the model presented in this paper, software programming is executed in four main activities (Fig. 1). Only the first activity is executed by programmers, while the remaining ones are system activities. A software architecture for BSNs has been developed to support these transparency levels [3].

This work presented a proposal for a paradigm shift in BSN programming. More suitable tools, such as programming interfaces and intelligent compilers, enable healthcare workers to become the actual programmers and maintainers of this technology. The possibility of developing and/or testing new applications without the need of specific technical knowledge on programming techniques and computational models can facilitate the popularization of this technology.

Intelligent compilers that are capable of embedding, during the programming at deployment time of the sensor nodes, mechanisms and policies that increase the effectiveness of the application, should also be targeted by new developments. These compilers should operate in a transparent way to the user, and must be based on the knowledge of specialists, since the medical expert could help in scheduling resources according to each application.

Future efforts will be concentrated on usability tests for measuring how well healthcare personnel can use this programming software model.

References

[1] G. Yang, *Body Sensor Networks*. London, England: Springer-Verlag, 2006.

[2] D. Baldus, K. Klabunde and G. Müsch, "Reliable set-Up of medical body-sensor networks", *Lecture Notes in Computer Science, 2920/2004*, 2004, pp.353-363.

[3] T. M. G. de A. Barbosa, I. G. Sene Jr, H. S. Carvalho, A. F. da Rocha, F. A. do Nascimento, and J. F. Camapum, J. F, "Application-oriented programming model for sensor networks embedded in the human body," in *Proc. of the 28th Annual International Conference IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 6037-6040, 2006.

[4] T. M. G. de A. Barbosa, I. G. Sene Jr, H. S. Carvalho, A. F. da Rocha, F. A. do Nascimento, and J. L. A. de Carvalho, "Programming Body Sensor Networks (Book chapter-Submitted for publication)", Encyclopedia of Healthcare Information Systems, submitted for publication.

[5] A. Lins, E. Nakamura, L. Rocha, A. A. F. Loureiro, and C. Coelho, "Semi-automatic generation of monitoring applications for wireless networks", in *Proc. of the* 9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'03), 2003.

[6] M. Welsh and G. Mainland, "Programming Sensor Networks Using Abstract Regions", in *Proc. of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04),* USENIX Press, *San Francisco*, 2004.

[7] J. Koshy and R. Pandey, "VM* synthesizing scalable runtime environments for sensor networks", in Proc. *3rd ACM Conference on Embedded Networked Sensor Systems*, ACM Press, *San Diego*, 2005.

[8] A. Pathak and V. K. Prasanna, "Issues in design a compilation framework for macroprogrammed networked sensor systems", in *Proc. 6th International Conference on Integrated Internet ad hoc and sensor networks*, ACM Press, Nice, 2006.

[9] H. T. Cormem, E. C. Leisersin and L. R. Rivest, *Algorithms.* MIT Press, Cambrige, MA, 1997.

[10] D. Dirky and C. Liechti, "The GCC toolchain for the Texas Instruments MSP430 MCUs" [Online]. Available: http://mspgcc.sourceforge.net/

[11] A. Sinha and A. Chandrakasan, "Dynamic Power Management in Sensor Networks (book chapter)", *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, CRC Press, 2004.

[12] Olimex. "MSP430-P149 MPS430F149 HEADER BOARD" [Online]. Available: http://www.olimex.com/dev/index.html

[13] Spark Fun. "Bluetooth Modem - BlueSMiRF Basic" [Online] Available:http://www.sparkfun.com/commerce/product_info.php?products_id=158